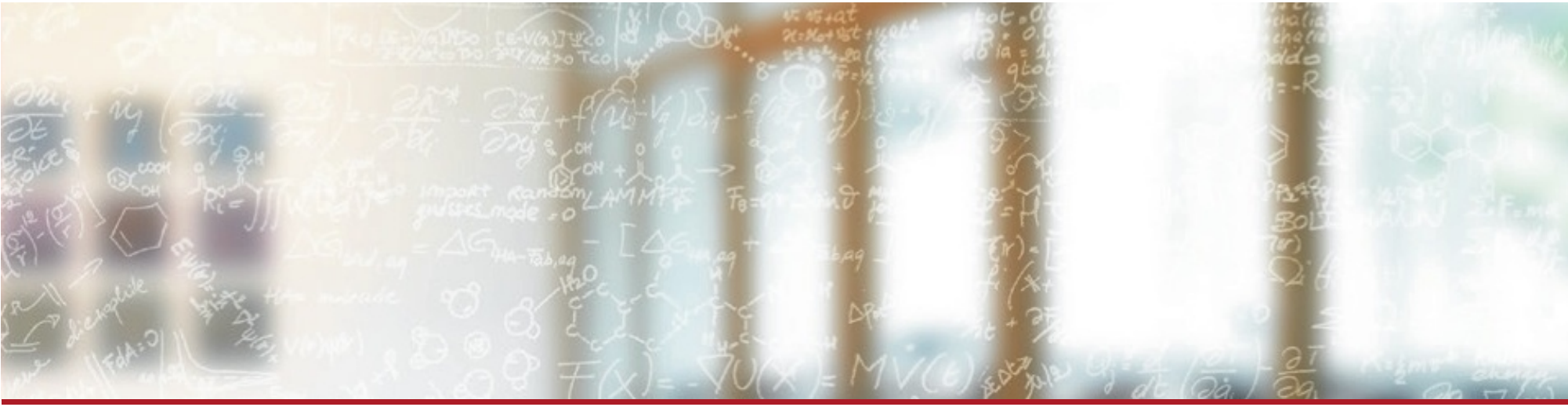


NestMC

A new multi-compartment neuronal network simulator

Alexander Peyser (FZ-J) & Sam Yates (CSCS)

November 3, 2016

NestMC

NestMC is a project to develop:

- a new multi-compartmental neuronal network simulator,
- that is optimized for HPC systems,
- and is easy to integrate into existing workflows.

See current development at

- `https://github.com/eth-cscs/nestmc-proto`

Who are we?

Cross-institutional collaboration

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

JÜLICH
FORSCHUNGSZENTRUM

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

As part of

Human Brain Project

Why?

Why develop a new simulator?

- There are problems and models that we can't explore with current software and systems.
- New HPC architectures.
- Adapting existing simulators to new architectures is *hard*.

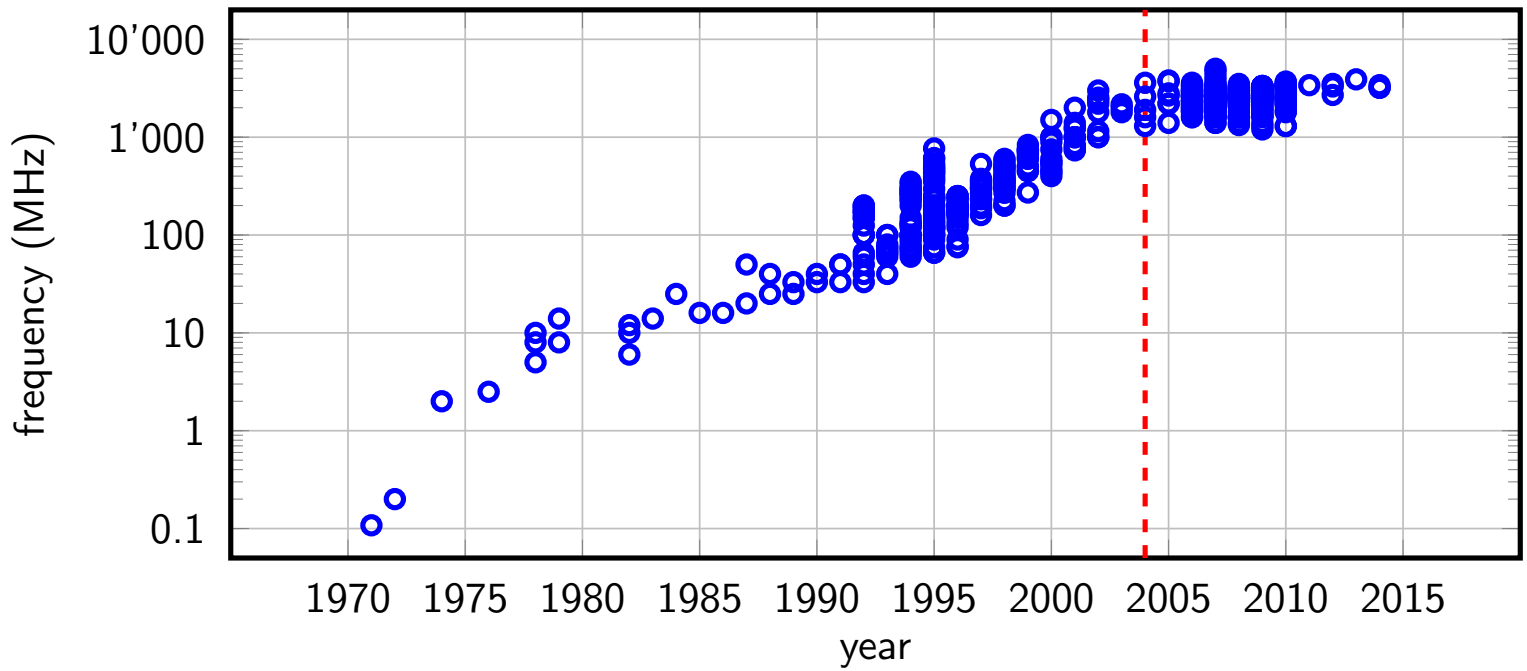
Hard problems

Examples

- Near real-time multi-compartment simulations.
- ‘Large’ networks:
  - long simulations,
  - parameter search,
  - statistical validation.
- Field potential calculations:
  - large networks,
  - volume visualization.

New architectures

Processor clock speed growth suddenly slowed around 2004.



Problem: power $\propto$ frequency^3

New HPC architectures

New performance gains primarily from:

- Highly parallel architectures (e.g. Intel KNL).
- Wider vector operations (e.g. AVX512).
- Specialized accelerator hardware: GPU, FPGA.

New HPC architectures

Prototype Human Brain Project HPC systems at Jülich

Julia

Intel many core KNL blade

Juron

IBM Power8+GPU 'fat' node

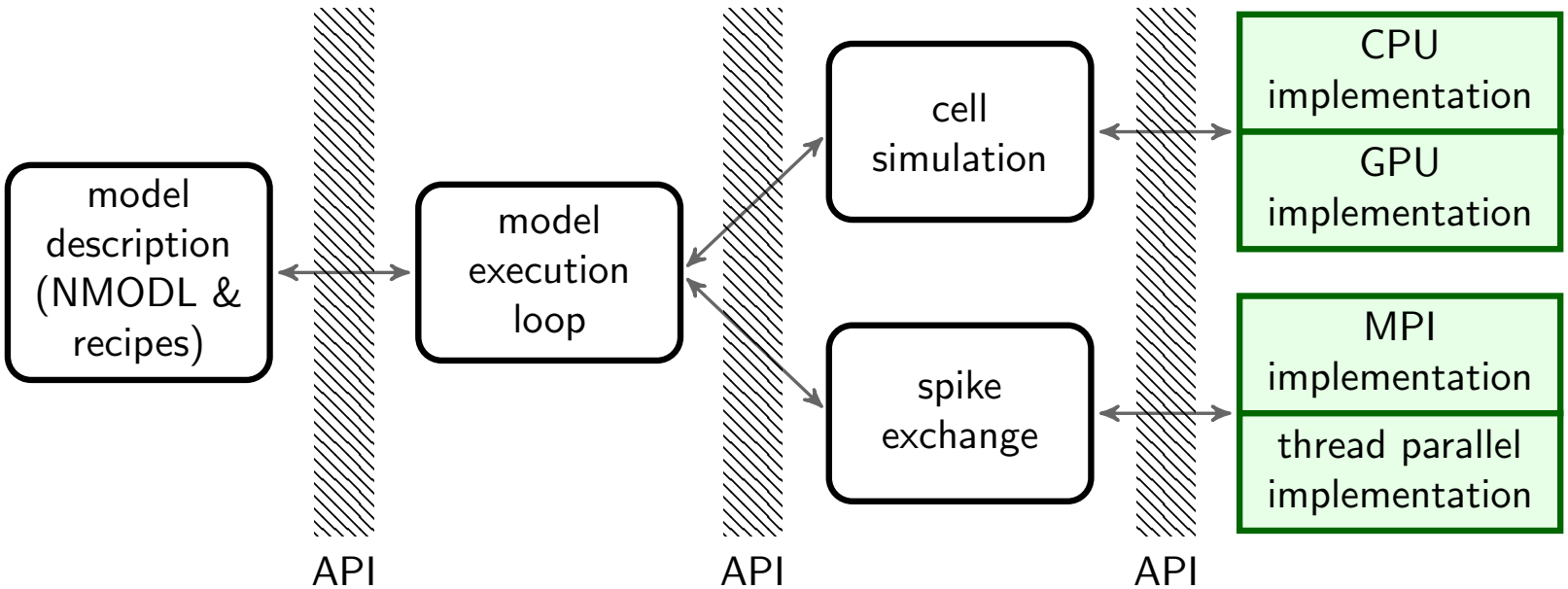
Efficient use demands new approaches.

We no longer get good performance improvement 'for free'.

Prototype design

Modular: components can be substituted according to internal API.

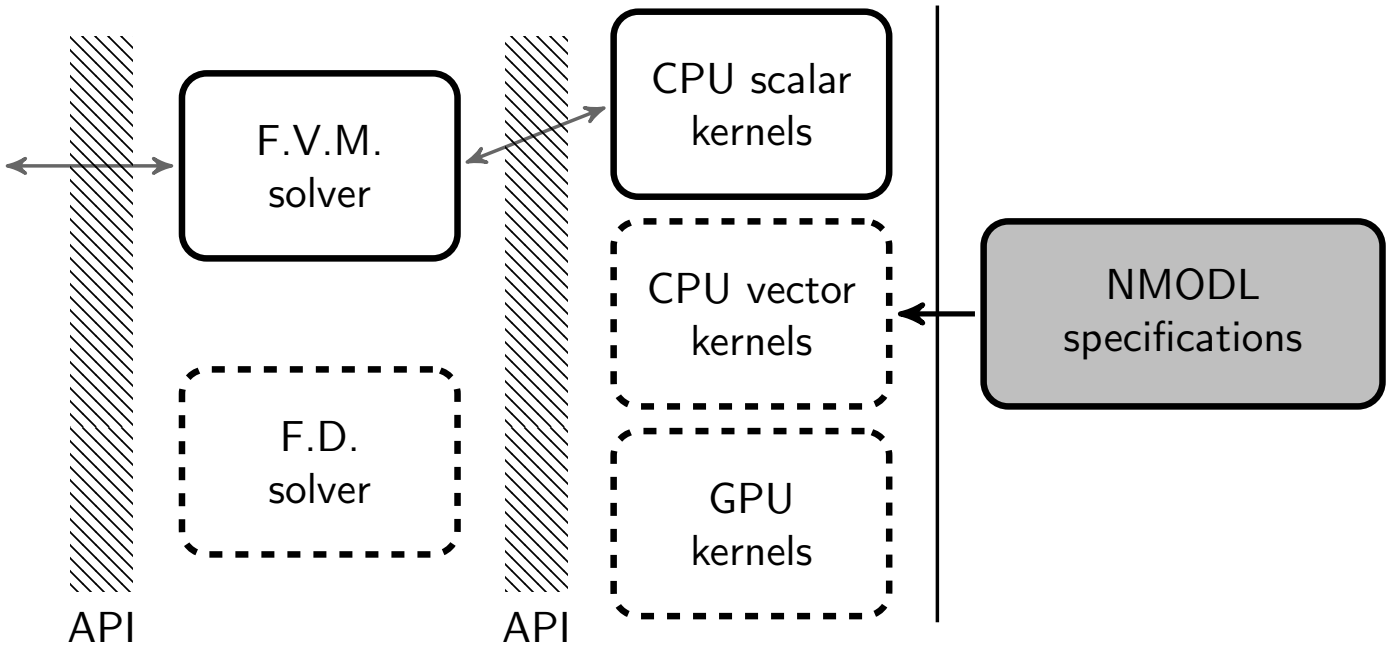
Internal API: ‘thin’ API; type parameterization allows components to determine low-overhead API data structures.



Prototype design — backends

Cell simulation modules share computational backends for channel and synapse state evolution.

CPU-hosted finite volume cell simulation



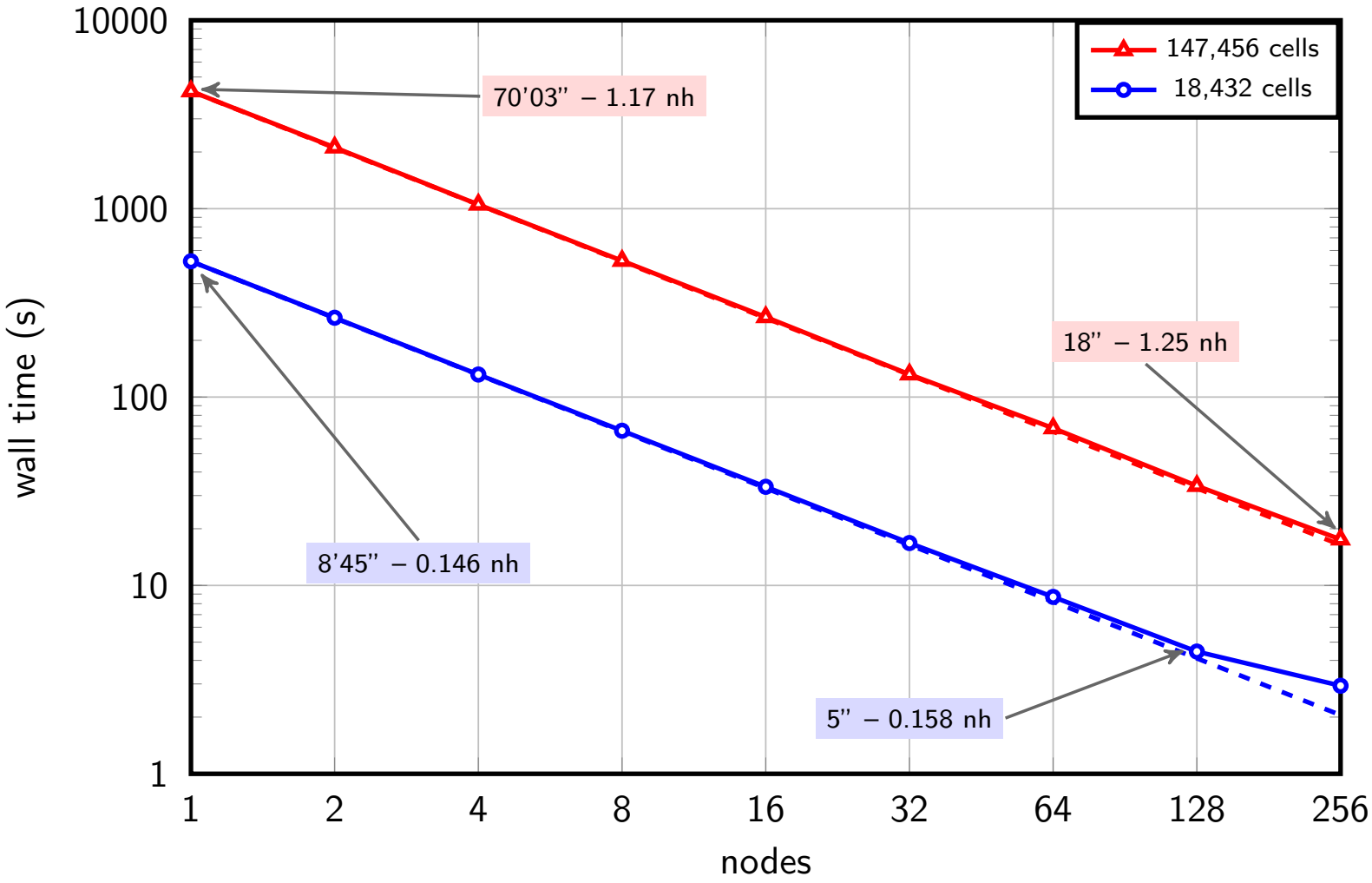
Prototype benchmarks

Test case

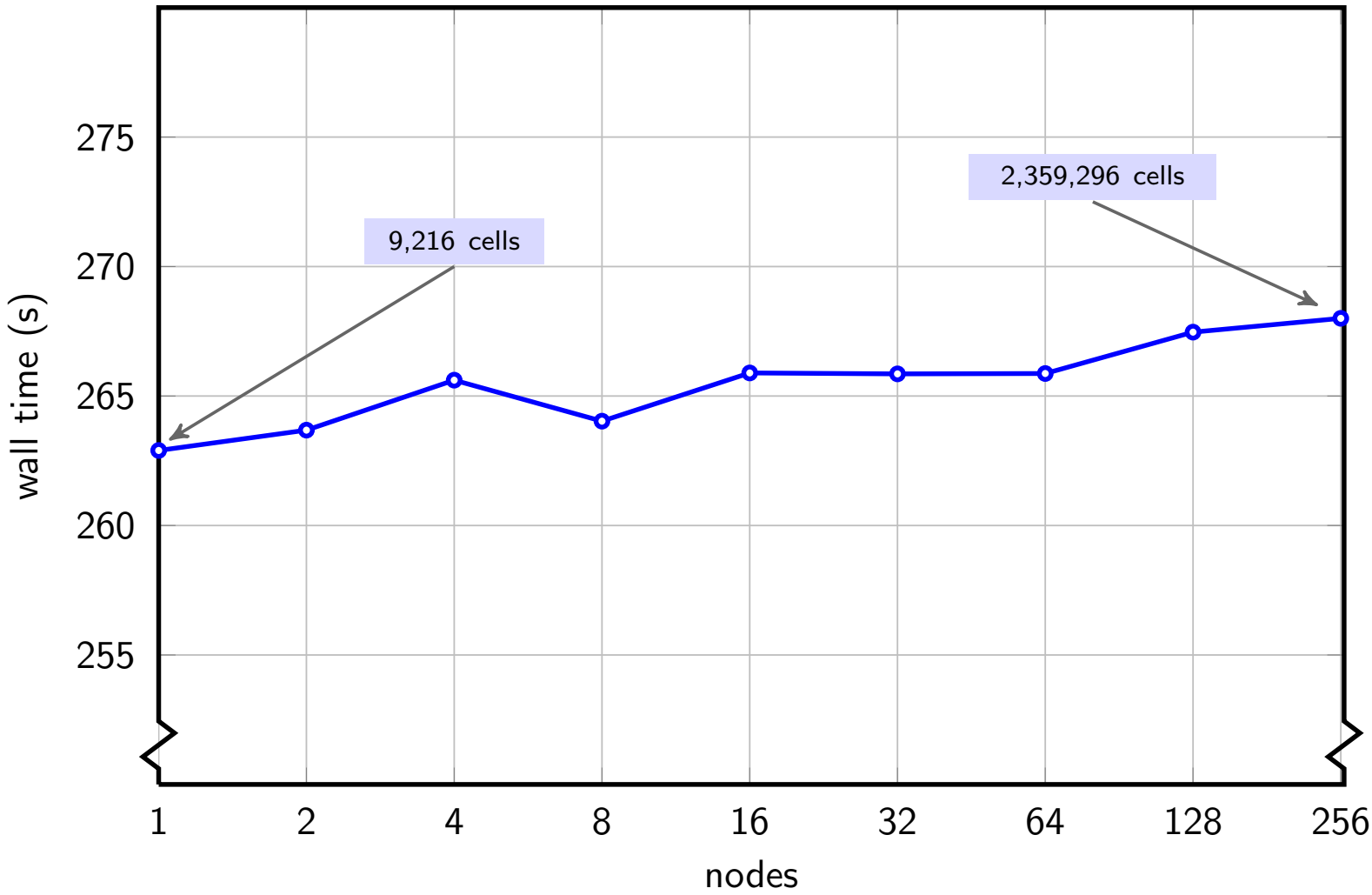
- 500 ms simulation.
- Each cell has 350 compartments and 2000 exponential excitatory synapses.
- H–H mechanism on cell somas, passive dendrites.
- Random network.
- Approximately 50 Hz spiking rate.

Benchmarks run on *Pitz Dora*, a Cray XC-40 system with 36 Broadwell cores per node.

Prototype benchmarks — strong scaling



Prototype benchmarks — weak scaling



Prototype status

Currently implemented

- Finite-volume based discretization.
- Distributed model instantiation.
- Spike and voltage trace output.
- x66 multi-core and Intel KNL support.
- Synapse and ion-channel descriptions in NMODL.
- Unit and validation testing suite.
- GPU support

How does this relate to NEST?

- Expertise: developers and experience from NEST goes into NestMC and what we learn from NestMC feeds back to NEST
- Infrastructure: Community and legal infrastructure can be leveraged for multiple products
- Interface: similar Python interfaces can reduce time for users to use multiple tools

How does this relate to NEST?

- Components: libraries such as for connectivity can be shared between projects
- Formats: commonality of formats and communications such as NESTml and I/O formats
- Multiscaling: hybrid simulations may be built across the spectrum from neural mass models to point models to compartment models...

How is this different from NEST?

- Models: solving point model ODEs is very distinct from large coupled compartment models
- Performance: NEST's point neurons are memory bound, NestMC is computationally bound
- Flexibility: NEST commits to maximum flexibility across platforms — NestMC is HPC focused on a subset of the "easy" 90% of cases
- Science: NEST is a highly reduced model for maximizing the size of simulations which are most amenable to mathematical analysis, while NestMC will be useful for morphologically detailed simulations

Thank you!

CSCS: Ben Cumming, Vasileios Karakasis, Stuart Yates
FZ-J: Wouter Klijn
BSC: Ivan Martinez

Contact bcumming@cscs.ch
a.peyser@fz-juelich.de

https://eth-cscs.github.io/nestmc
https://github.com/eth-cscs/nestmc-proto

Participate

Success depends on facilitating use cases!

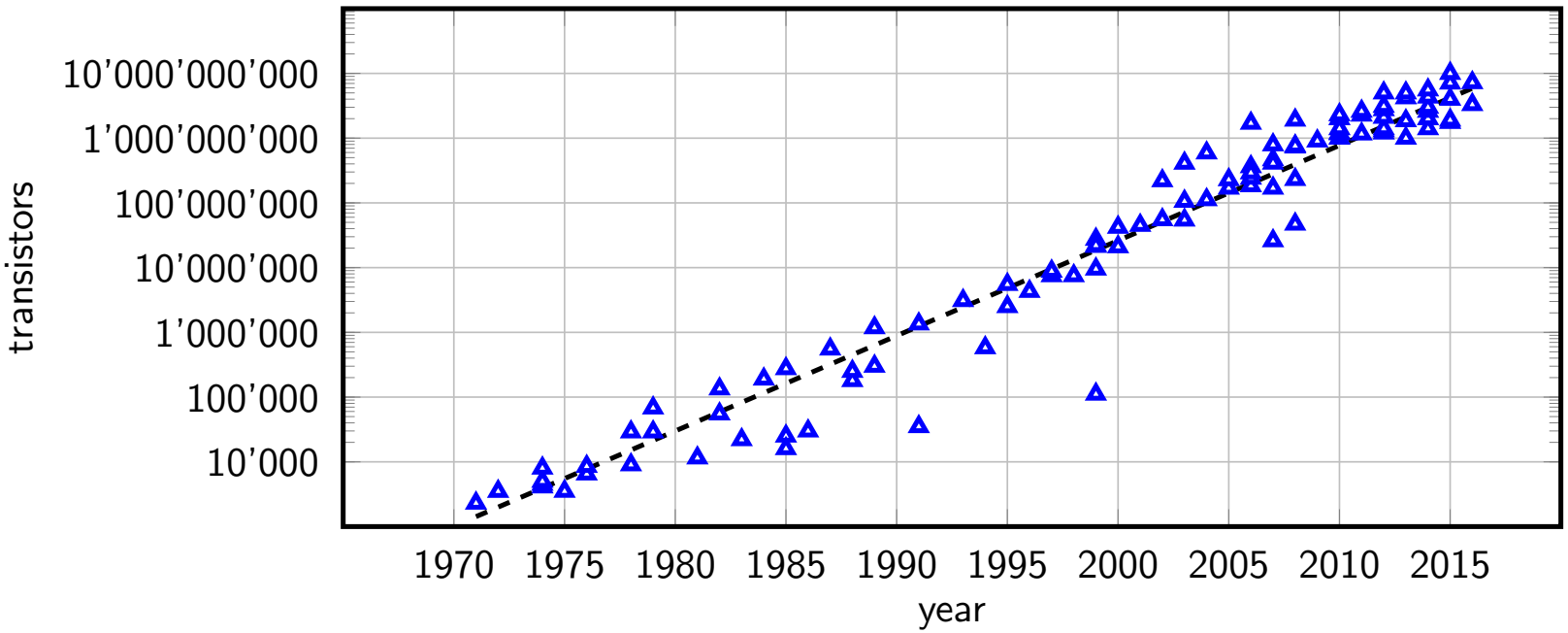
Do you have a use case which is hard to simulate with current tools?

Are there computational experiments that you wish to run, but currently cannot?

We want to work closely with researchers and research groups to ensure that our designs meet real needs in the community.

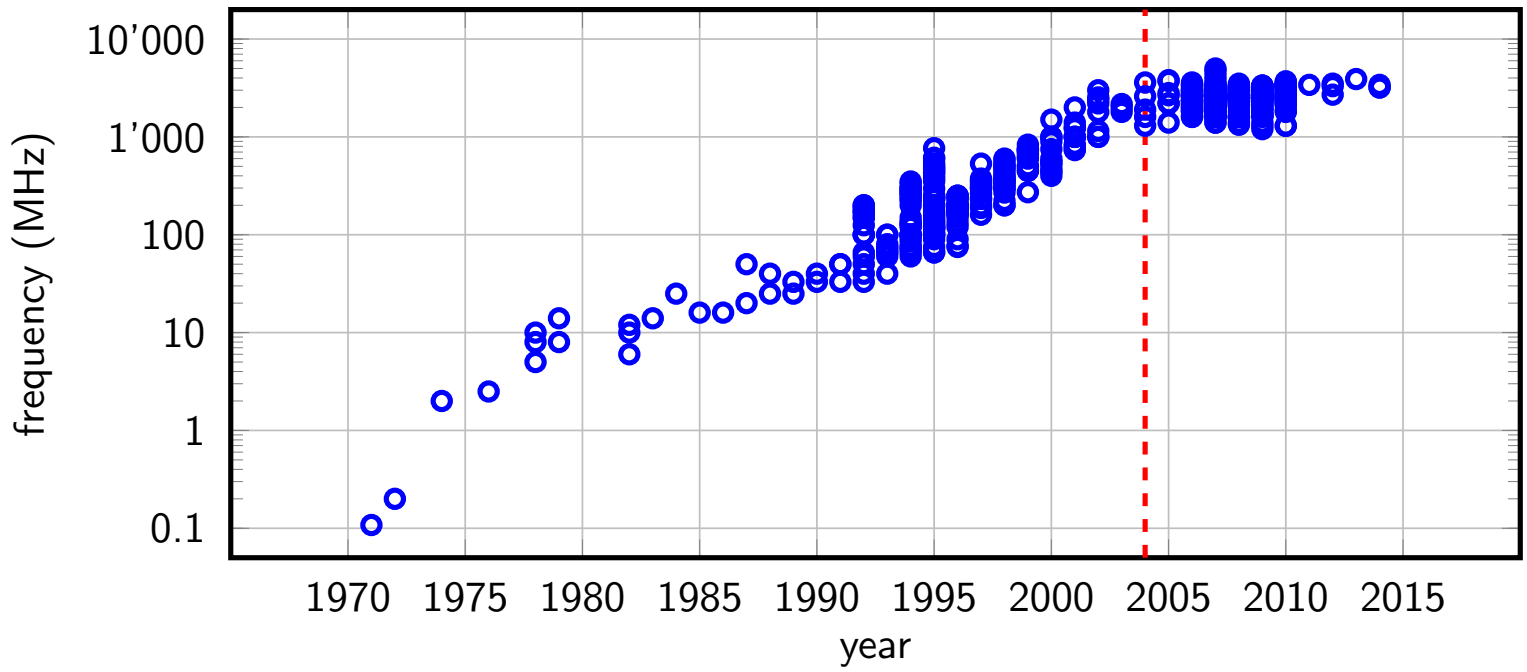
The limits of frequency

Microprocessor transistor counts have grown exponentially over a very long time frame.



The limits of frequency

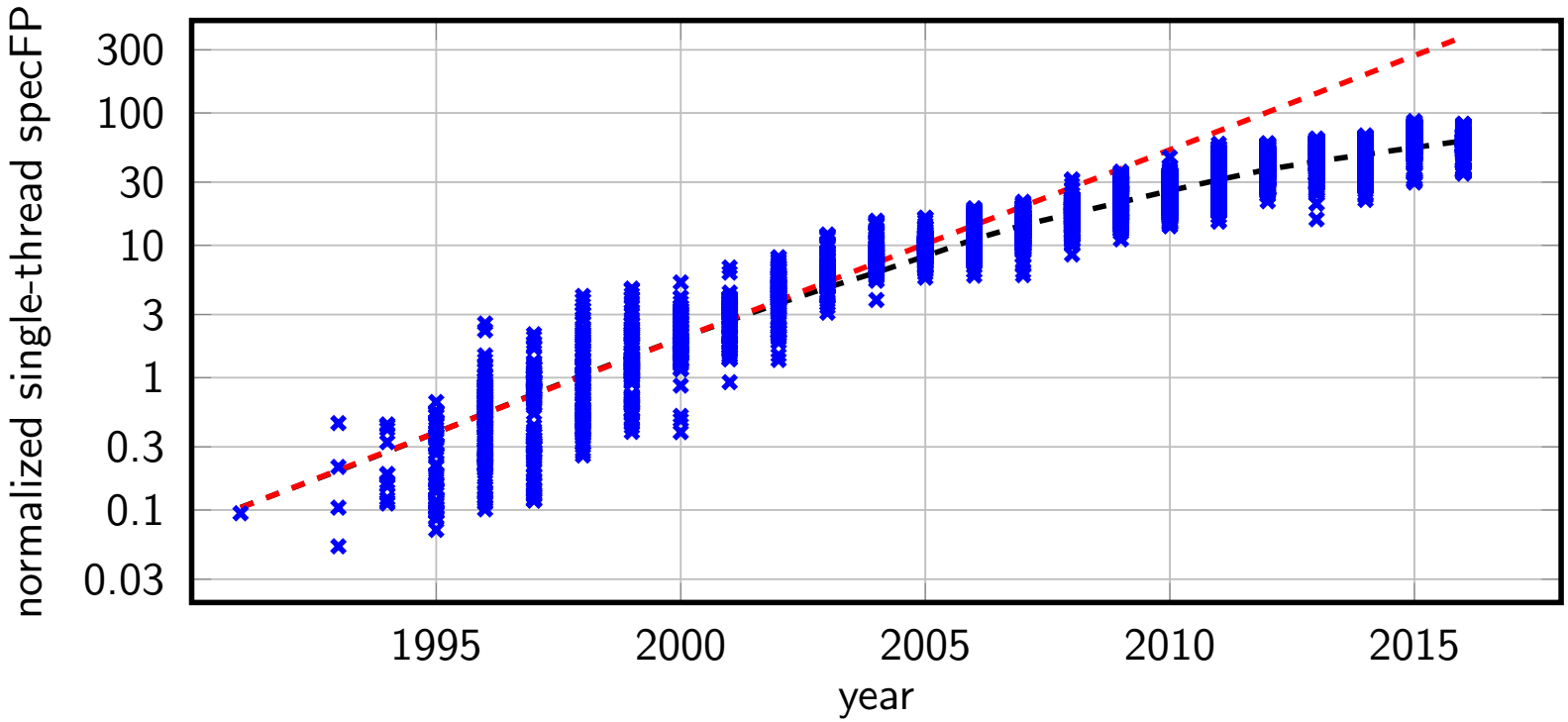
Processor clock speed growth suddenly slowed around 2004.



Problem: power $\propto$ frequency^3

The limits of frequency

Single-thread performance growth also slows.



Analysis thanks to Jeff Preshing, http://preshing.com

Why?

Why develop a new simulator?

- There are problems and models that we can't explore with current software and systems.
- New HPC architectures.
- Adapting existing simulators to new architectures is *hard*.

Existing simulators

NEURON and GENESIS have had a very long development. NEURON in particular is very large with many features.

Newer simulators such as MOOSE and Brian still primarily target the workstation.

GPU support for these simulators still under development.

Adapting existing large applications to highly parallel and hardware-accelerated architectures is non-trivial.

Opportunity

A new development project can:

- target contemporary and future HPC architectures,
- be co-designed to faciliatate new and difficult use cases.

Opportunity

A new development project can:

- target contemporary and future HPC architectures,
- be co-designed to faciliatate new and difficult use cases.

In addition,

- much easier to adopt modern software development processes from the start.

NestMC

Two year initial project to design and develop a multi-compartmental simulator for HPC systems.

Goals

Interoperability

Simulator as library

- Visualization
- Multi-physics
- Multi-scale

Extensibility

Modular internal API

- New integration schemes
- Custom spike communication
- Specialized cells

Performance

HPC targeted

- Highly parallel
- GPU and vector targets
- Design for scalability

Development timeline

Now

Prototype development.

12/2016

Wind-up prototype.
Finalize design of initial release.

4/2017

First public release of simulator.
Move to open development model.

Prototype

The prototype development allows us to explore the design space.

“Plan one to throw one away” — Fred Brookes

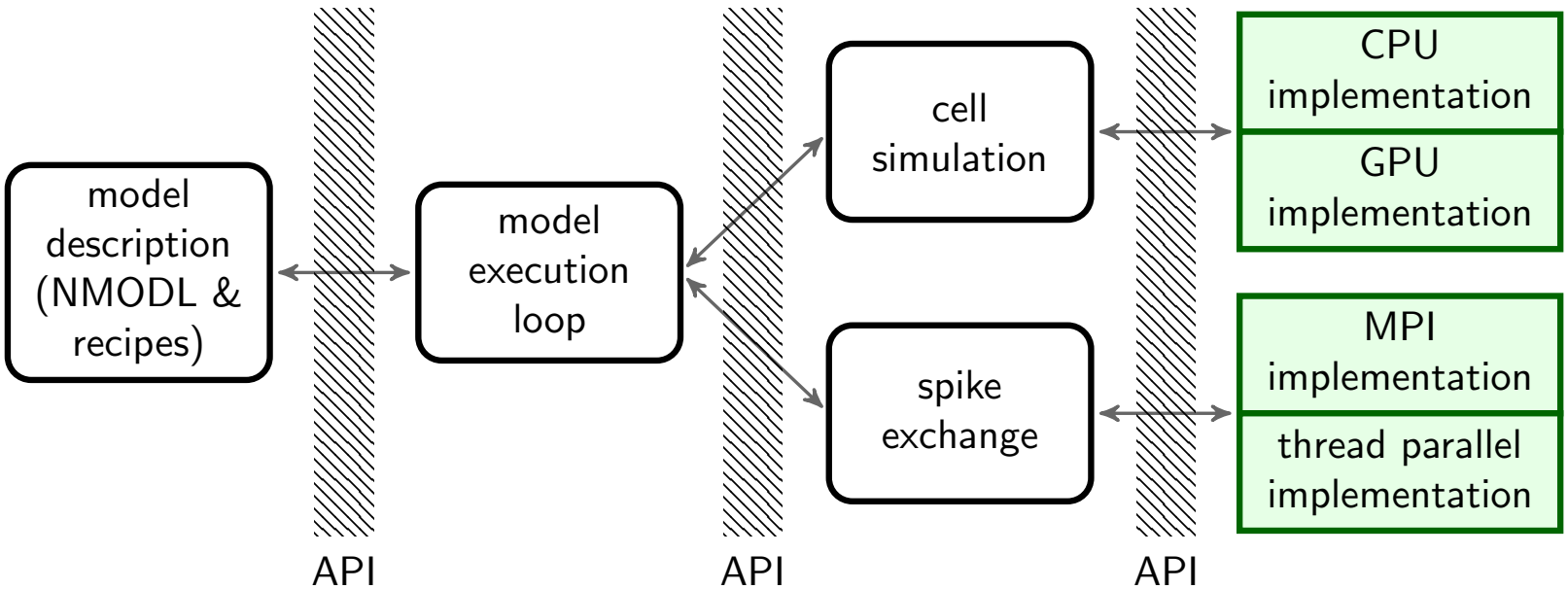
Currently implementing features and use cases to refine our design:

- LFP live visualization $\rightarrow$ interoperability features.
- Gap junctions $\rightarrow$ extensibility, internal API design.
- GPU execution $\rightarrow$ performance, modularity.

Prototype design

Modular: components can be substituted according to internal API.

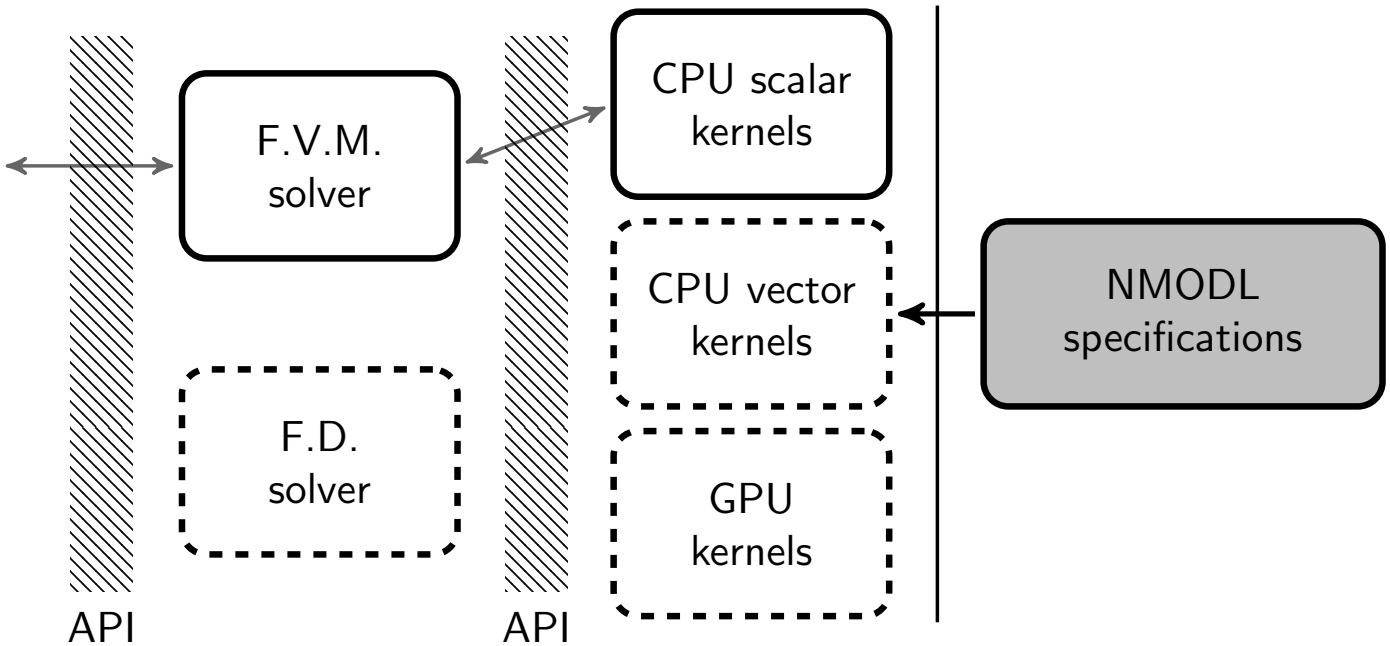
Internal API: ‘thin’ API; type parameterization allows components to determine low-overhead API data structures.



Prototype design — backends

Cell simulation modules share computational backends for channel and synapse state evolution.

CPU-hosted finite volume cell simulation



Prototype observations

Design

- Abstractions over communication and threading
  $\rightarrow$ greatly simplified testing and validation.
- Component architecture
  $\rightarrow$ rapid prototyping,
  $\rightarrow$ use-case driven API changes are limited in scope.
- Functional model description
  $\rightarrow$ reproducible across different systems,
  $\rightarrow$ distributed instantiation.

Prototype observations

Development practices

- Unit and validation testing
  $\rightarrow$ limits exposure to bugs and design errors.
- Large matrix of compilers and hardware targets
  $\rightarrow$ continuous integration a necessity
- ‘Agile’ iterative refinement of development practices allows us to adapt our processes to our team distributed across multiple institutions and countries.